

레이어제로(LayerZero)

해당 번역본은 이용자가 디지털자산 관련 정보를 편리하게 확인할 수 있도록
단순 참고용으로 번역된 것이며, 투자 권유를 목적으로 하지 않습니다.
그 내용이 정확하지 않을 수 있으므로 정확한 정보 습득을 위해서는
원문을 참고하시거나 원문 작성 주체 측에 문의하시기 바랍니다.
두나무는 해당 번역본 내용의 진실성 및 정확성을 보장하지 않고,
제공된 정보에 의한 투자 결과에 대하여 어떠한 책임도 지지 않으며,
투자 손실은 투자자에게 귀속됩니다

LayerZero

라이언 자리크
브라이언 펠레그리노
아이작 장
토마스 킴
캘럽 배니스터

LayerZero Labs

초록

본 논문에서는 최초의 *본질적으로 안전하며 의미론적으로 보편적인* 옴니체인 상호운용성 프로토콜인 LayerZero를 제시한다. *불변의* 엔드포인트, 추가 전용 검증 모듈, 완전하게 구성 가능한 검증 인프라를 활용하여 LayerZero는 옴니체인 상호운용성을 달성하기 위해 필요한 보안성, 구성 가능성, 확장성을 제공한다. LayerZero는 모든 블록체인과 사용 사례를 보편적으로 지원하도록 설계된 새로운 신뢰 최소화 모듈형 보안 프레임워크를 통해 프로토콜 보안과 비용에 대해 애플리케이션 전용의 엄격한 소유권을 부여한다. LayerZero 프로토콜 위에서 구축된 옴니체인 애플리케이션(OApps)은 LayerZero의 보편적 네트워크 의미론을 통해 블록체인 불가지론적이고 매끄러운 상호 운용을 달성한다.

1 서론

블록체인 상호운용성은 체인의 다양성이 지속적으로 증가함에 따라 점점 더 큰 도전 과제로 떠오르고 있으며, 애플리케이션이 점점 더 많은 체인에서 사용자에게 도달하려 함에 따라 단편화된 블록체인 생태계를 연결하는 중요성이 높아지고 있다. 우리는 모든 블록체인과 사용 사례에 확장 가능한 완전 연결 메시 네트워크를 달성한 최초의 옴니체인 메시징 프로토콜(OMP)인 LayerZero를 제시한다.

다른 크로스체인 메시징 서비스의 단일적 보안 모델과 달리, LayerZero 프로토콜은 보안을 *불변하게* 구현하기 위해 새로운 모듈형 보안 모델을 사용한다. 이러한 접근 방식은 여전히 새로운 기능과 검증 알고리즘을 지원하도록 확장될 수 있다. 검열, 재생 공격, 서비스 거부 공격, 그리고 코드의 현장 수정에 대한 *본질적 보안*은 *불변의* 엔드포인트에 설계되어 있다. 서명 방식 등과 같은 덜 근본적이거나 *외재적*인 보안 요소들은 독립적으로 불변인 모듈로 분리된다. 프로토콜로서 LayerZero는 특정 인프라나 블록체인에 종속되지 않으며, 엔드포인트를 제외한 모든 구성 요소는 LayerZero 위에 구축된 애플리케이션에 의해 교체되고 구성될 수 있다.

그림 1에서 옴니체인 완전 연결 메시 네트워크를 설명한다. 각 체인은 다른 모든 체인과 직접 연결되어 있으며, *외재적* 보안(2.1절)은 체인 쌍마다 다를 수 있지만(색깔 있는 실선으로 표시됨), 최종적으로 손실 없이 정확히 한 번만 패킷이 전달된다는 보장은 항상 동일하며 변하지 않는다.

LayerZero의 네트워크 채널 의미론은 실행 기능, 구성 의미론, 검열 저항성, 장애 모델을 포함하여 보편적이다. 이러한 보편적 의미론은 애플리케이션 개발자가 안전하고 체인 불가지론적인 옴니체인 애플리케이션(OApps)을 손쉽게 설계할 수 있도록 한다.

Cross-chain

Omnichain

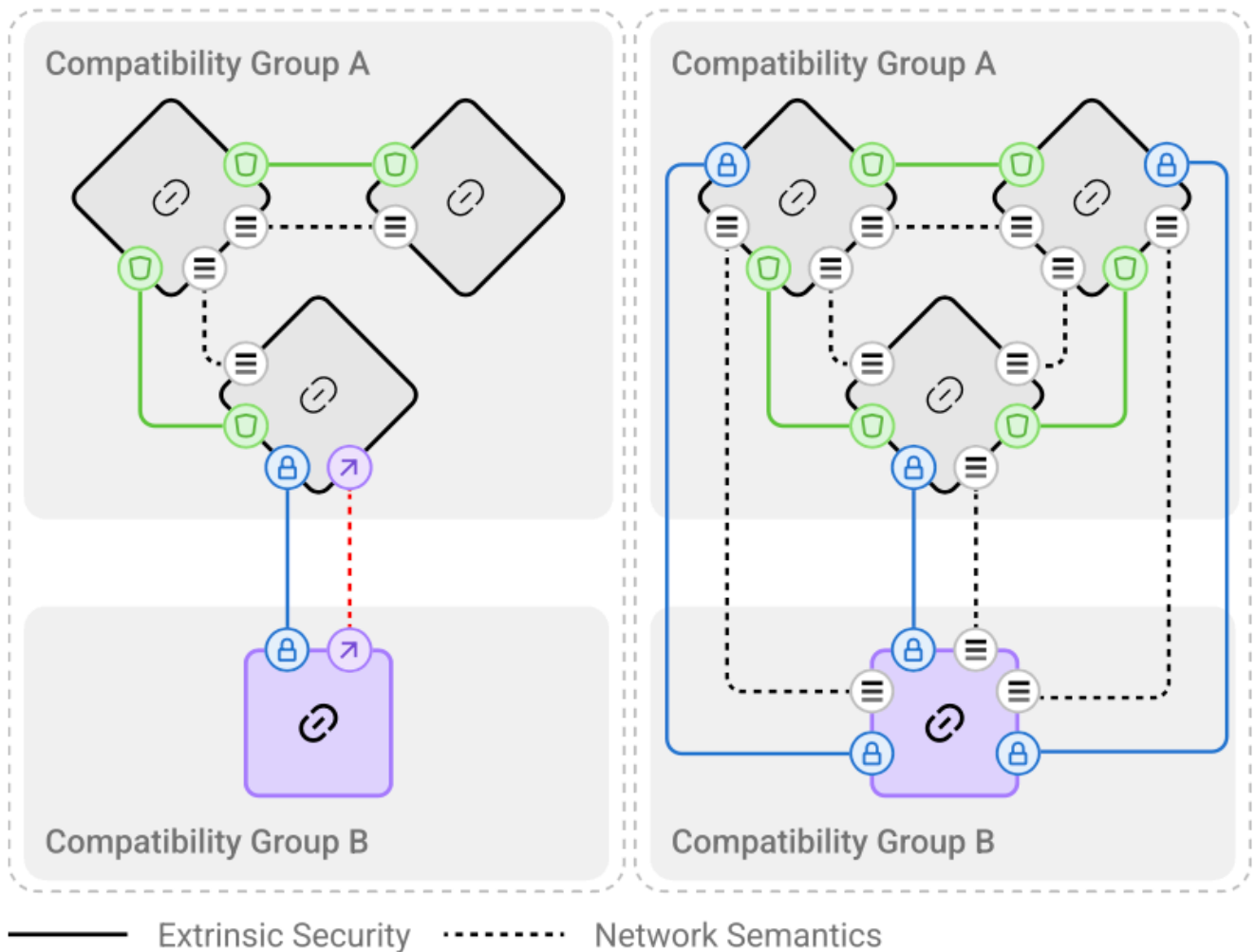


Figure 1: The omnichain fully-connected mesh network has universal network semantics for all connected chains and security specialized to each link.

본 논문의 나머지 부분은 다음과 같이 구성된다.

먼저, 2장은 LayerZero 프로토콜의 근본적인 원칙을 설명한다. 3장은 프로토콜 설계를 다루며 각 구성 요소가 보안을 위해 어떻게 구조화되었는지를 강조한다. 마지막으로, 4장은 LayerZero가 블록체인 불가치론적 방식으로 광범위한 추가 기능을 쉽게 지원하도록 확장될 수 있는 사례들을 제시한다.

2 원칙

OMP의 책임은 본질적 보안과 보편적 의미론이라는 두 가지 요구사항으로 압축될 수 있다. 기존의 메시징 서비스들은 위 요구사항 중 하나 또는 둘 다를 구현하지 못하며, 이로 인해 단일적 보안과 과도한 특수화라는 두 가지 근본적인 결함을 겪는다. 본 절의 나머지 부분에서는 크로스체인 메시징 패러다임 내에서 보안과 의미론을 맥락화하고, 기존 크로스체인 메시징 시스템들이 이러한 목표에 어떻게 미치지 못하는지 설명하며, LayerZero가 이러한 한계를 극복하기 위해 어떻게 근본부터 설계되었는지를 개괄한다.

2.1 보안

OMP의 첫 번째이자 가장 중요한 요구사항은 안전해야한다는 점이다. 우리는 보안을 *본질적* 보안과 *외재적* 보안으로 구분하며, 모든 메시징 시스템이 외재적 보안을 구현하지만, 본질적 보안을 제공하는 경우는 드물다. 본질적 보안은 무손실(검열 저항성), 정확히 한 번(재생 없음), 최종적(생존성) 전달이라는 프로토콜 수준의 불변성을 의미한다. 외재적 보안은 서명 및 검증 알고리즘과 같은 그 밖의 모든 보안 속성을 포괄한다. 대부분의 기존 메시징 서비스들은 보안에 대해 임시방편적 접근을 취하여, 네트워크에 체인이 추가될 때마다 이를 수용하기 위해 단일적 엔드투엔드 보안 모델을 지속적으로 갱신해왔다. 이러한 서비스들은 필연적으로 공유된 보안 모델에 강제적인 현장 업데이트를 사용하므로, OApps가 기반으로 삼을 수 있는 장기적 보안 불변성을 제공할 수 없다. LayerZero에서는 장기적 보안 불변성을 제공하기 위해 보안을 모듈화하고, 모든 모듈에 대해 엄격한 불변성을 적용하는 방식을 선택했다.

무결성 계층	장애 모델
채널 유효성	패킷 검열
	패킷 재생
	결함 있는 업데이트
	잘못된 재구성
채널 생존성	서비스 거부
	인프라 상태
	관리자 상태
데이터 유효성	암호학적 공격
	악의적인 인프라
데이터 생존성	블록체인에서의 데이터 손실

표 1: 우리는 프로토콜 무결성을 네 가지 속성으로 구분한다. 데이터 생존성은 기반 블록체인에 의존하며 OMP에 의해 보장될 수 없다.

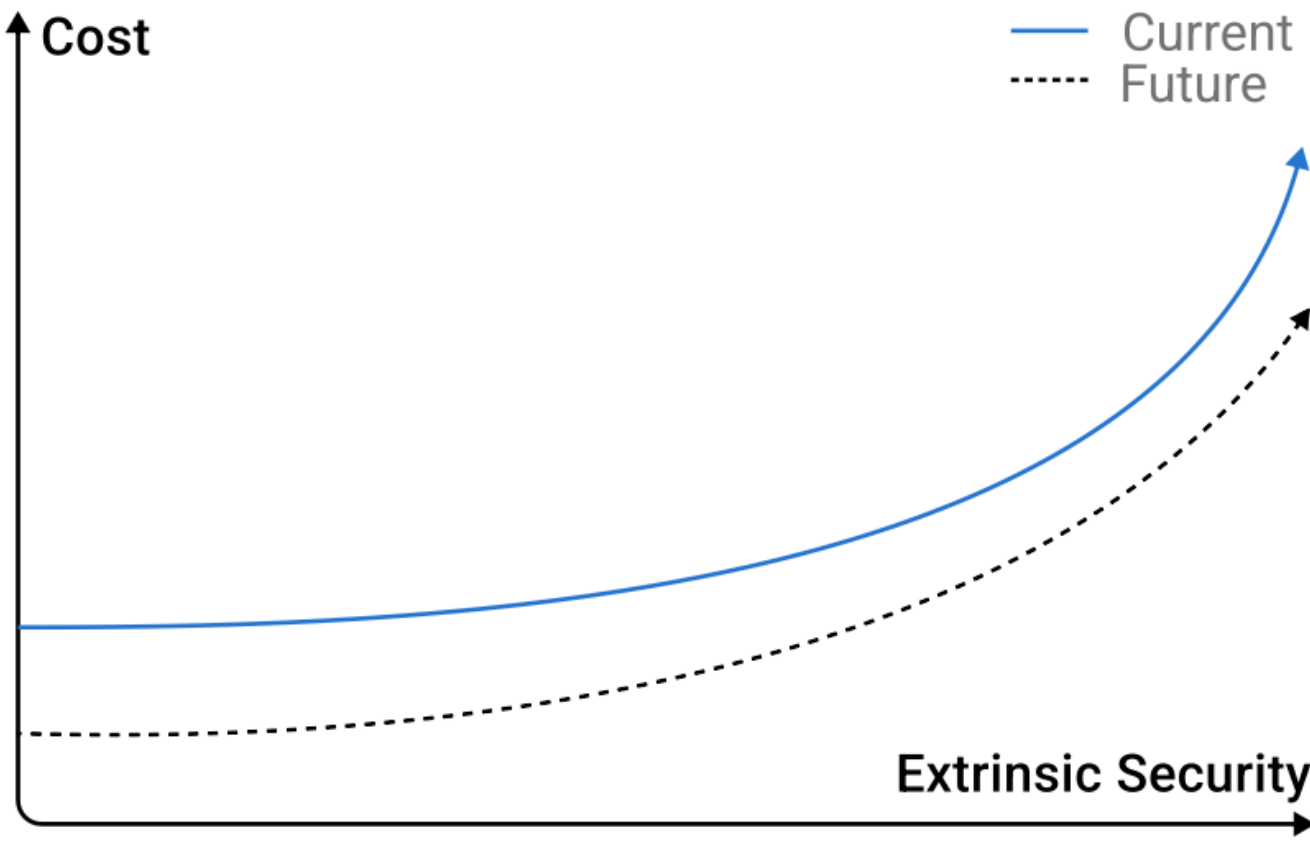


Figure 2: The pareto frontier of extrinsic security vs cost continually changes due to advancements in technology.

표 1은 우리가 프로토콜 무결성을 채널 무결성과 데이터 무결성으로 구분하는 방식을 보여준다. 이러한 무결성 계층 각각은 다시 유효성과 생존성 속성으로 세분된다. 우리는 본질적 보안을 채널 유효성과 생존성을 포괄하는 것으로, 외재적 보안을 데이터 유효성을 포괄하는 것으로 보다 형식적으로 정의한다. 본 논문에서는 외재적 보안 구성을 *보안 스택(Security Stack)*이라 지칭한다. 단일적 공유 보안 시스템은 모든 애플리케이션에 동일한 보안 스택을 강제하는 반면, 분리된 보안 시스템은 OApp별로 서로 다른 보안 스택을 허용한다.

본질적 보안은 제일 원칙(first principles)에 기반하여 보편적으로 보장될 수 있고, 또 그래야 한다. 반대로, 블록체인 간 최적의 신뢰 없는 통신은 불가능하며, 검증 알고리즘과 블록체인 설계의 지속적인 발전은 외재적 보안의 확장성과 구성 가능성을 필연적으로 요구한다. 이는 L1-L2 롤업과 같은 특수한 경우에도 마찬가지이다. 하드포크 가능성은 L2 컨트랙트의 업그레이드 가능성을 필요로 하며, 따라서 L2 컨트랙트 소유자를 신뢰할 수밖에 없는 주체로 만든다.

다층의 실행 환경과 NP 문제의 계산 불가능성에 의존하는 암호 알고리즘을 고려할 때, 비사소한 코드의 외재적 보안을 형식적으로 검증하는 것은 실행 불가능하다. 결과적으로 가장 실질적인 보안 측정 기준은 경제적 기준이다. 즉, 업그레이드 불가능한 스마트 컨트랙트의 보안은 그것이 얼마나 많은 자산을 얼마 동안 보호했

는지에 정비례한다. 따라서 외재적 보안의 분리(isolation)가 가지는 시사점은 명확하다. OMP는 확립된 외재적 보안을 가진 코드에 무기한 접근을 보장하면서도, 프로토콜 관리자가 프로토콜을 확장할 수 있도록 허용해야 한다.

2.2 보편적 의미론

OMP의 두 번째 요구사항은 보편적 의미론, 즉 네트워크 기본 요소를 모든 추가적인 사용 사례와 블록체인에 확장하고 적응시킬 수 있는 능력이다.

실행 의미론(즉, 기능 로직)은 체인 불가지론적이어야 할 뿐만 아니라, 어떠한 OApp이 요구하는 기능도 가능하게 할 수 있을 만큼 충분히 표현력이 있어야 한다. 우리가 LayerZero를 설계하면서 얻은 핵심 통찰은 기능 로직(실행)을 보안(검증)으로부터 완전히 분리할 수 있다는 점이다. 이는 프로토콜 개발을 단순화했을 뿐만 아니라, *실행* 기능을 설계하고 구현할 때 프로토콜 보안에 미치는 영향에 대한 우려를 제거한다.

보편적 의미론의 또 다른 측면은 OMP 인터페이스와 네트워크 의미론이 모든 기존 및 미래 블록체인과 보편적으로 호환되어야 한다는 점이다. 의미론적 통합의 중요성은 아무리 강조해도 지나치지 않다. 네트워크에 추가되는 모든 블록체인마다 서로 다른 인터페이스와 네트워크 일관성 모델을 수용하기 위해 상당한 엔지니어링 비용이 발생한다면 OApps는 확장될 수 *없기* 때문이다. 실제로, OMP는 출발지 및 목적지 블록체인의 특성과 무관하게 통합된 인터페이스, 전송 의미론, 실행 동작을 가져야 한다.

3 핵심 프로토콜 설계

우리는 LayerZero를 네 가지 구성 요소로 구분한다(그림 3). 검열 저항성을 구현하는 불변의 *엔드포인트* (Endpoint), 온체인 검증 모듈(MessageLibs)의 추가 전용 모음인 *MessageLib 레지스트리* (MessageLib Registry), 블록체인 간 데이터를 검증하기 위한 탈중앙화 검증자 네트워크(DVNs)의 무허가 집합, 그리고 크로스체인 메시지 검증 맥락과 분리된 상태에서 기능 로직을 실행하는 무허가 *실행자(executors)*이다. 이 모든 구성 요소를 연결하는 것은 *OApp 보안 스택(Security Stack)*으로, 이는 프로토콜의 외재적 보안 구성을 정의하며 오직 OApp 소유자만 수정할 수 있다.

LayerZero의 메시지는 페이로드와 라우팅 정보(경로)로 구성된다. 이러한 메시지는 메시 네트워크를 통해 전송되기 전에 *패킷*으로 직렬화된다. 패킷은 대상 블록체인의 검증 계층에 의해 검증된 후 무손실 채널에 *커밋(commit)*된다. 이후 패킷은 채널에서 읽혀지고, 대상 OApp 컨트랙트의 *lzReceive* 콜백을 실행함으로써 *전달(deliver)*된다.

*LayerZero 엔드포인트(Endpoint)*는 OApp 전용 보안 스택(Security Stack) 소유권과 함께, 검열 저항성, 정확히 한 번 전달, 보장된 생존성을 구현하는 불변의 채널을 통해 채널 유효성을 보호한다. 엔드포인트의 불변성은 어떠한 외부 주체나 조직도 OApp의 보안 스택 보안 특성을 강제로 변경할 수 없음을 보장한다.

개별적으로 불변인 MessageLibs는 MessageLib 레지스트리에서 선택할 수 있으며, 각 MessageLib는 패킷 데이터 무결성을 검증한 후에야 메시지가 엔드포인트에 커밋될 수 있도록 허용하는 외재적 보안 인터페이스이다.

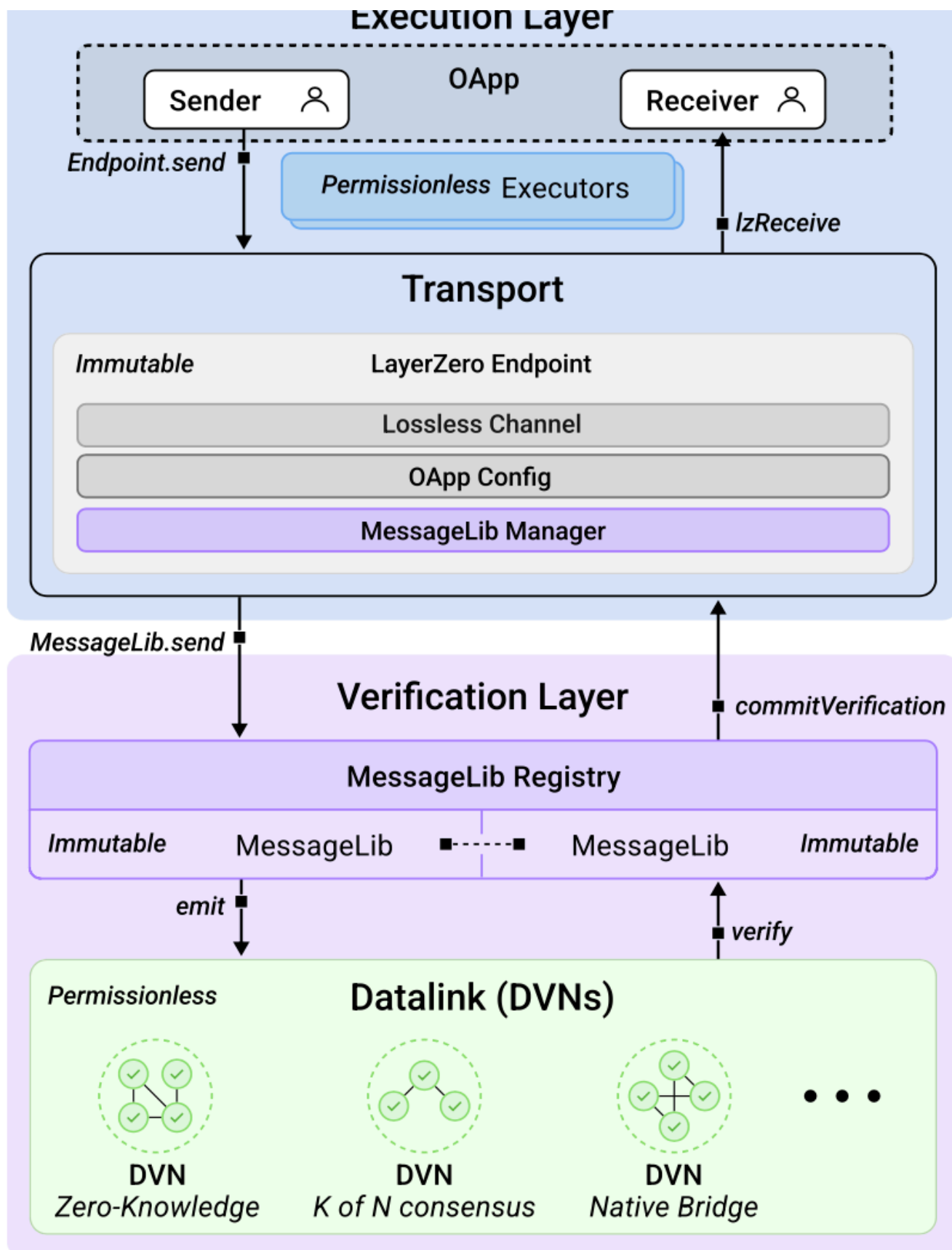


Figure 3: LayerZero is divided into execution and verification layers. The verification layer securely transmits data between blockchains, and the execution layer inter-

prets this data to form a secure, censorship resistant messaging channel.

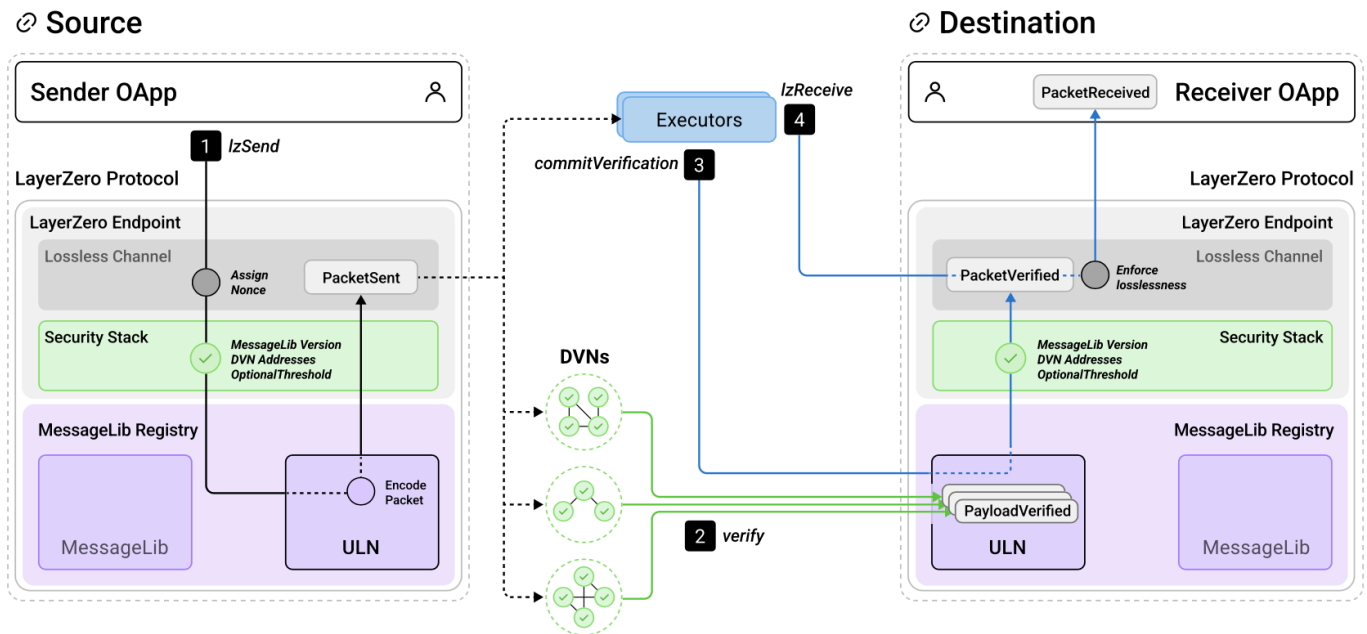


Figure 4: Steps to send a message using LayerZero.

기존 *MessageLibs*는 수정될 수 없으므로 *MessageLib 레지스트리*는 추가 전용 구조가 되며, 각 *보안 스택(*Security Stack*)*은 정확히 하나의 *MessageLib*를 지정한다. 보안 스택 소유권 의미론과 *MessageLib*의 불변성이 결합되어, 애플리케이션은 잠재적으로 동일한 보안 스택을 영구적으로 사용할 수 있다.

각 DVN은 두 독립된 블록체인 간에 공유되는 데이터의 무결성을 집합적으로 검증하는 *검증자 (verifiers)*들의 집합이다. DVN은 오프체인 및/또는 온체인 구성 요소를 포함할 수 있으며, 각 보안 스택은 이론적으로 무한한 수의 DVN을 포함할 수 있다. 기본적인 DVN 구조는 영지식 증명, 사이드체인, K-of-N 합의, 네이티브 브리지 등을 포함하되 이에 국한되지 않는 어떤 검증 메커니즘도 활용할 수 있다. 간결성을 위해, 우리는 *MessageLib*, DVN, 그리고 기타 하이퍼파라미터들의 추상적 집합을 *보안 스택 (*Security Stack*)*이라 부르며, 보안 스택의 직렬화된 형식이 각 체인에 기록되는 것으로 가정한다.

채널 생존성(최종적 전달)은 *무허가 실행(permissionless execution)*과 보안 스택(*Security Stack*) 재구성을 통해 보장된다. 출발지와 목적지 블록체인이 생존성을 유지한다고 가정할 때, *LayerZero* 채널 생존성이 (일시적으로) 손상될 수 있는 경우는 (1) 보안 스택 내 DVN이 릴레이를 만료하거나 장애를 일으킬 때, (2) 구성된 실행자가 메시지 전달을 중단할 때뿐이다. 구성된 DVN이 너무 많이 메시지 검증을 중단하면, OApp은 보안 스택을 다른 DVN으로 재구성하여 생존성을 회복할 수 있다. 패킷 전달(실행)은 무허가 방식이므로, 실행 가스 비용을 지불할 의사가 있는 모든 주체가 패킷을 전달하여 채널 생존성을 복구할 수 있다.

LayerZero 구성 요소 간 상호작용은 최소화되고 표준화되어 소프트웨어 버그 발생 표면을 줄인다(그림 3). 또한 LayerZero의 모듈화와 구성 가능성은 새로운 체인에서 프로토콜의 신속한 프로토타이핑을 가능하게 한다. MessageLib의 자리표시자로 단순한 화이트리스트를 사용하는 방식은 모든 구성 요소의 병렬 개발 및 테스트를 허용하여 메시 네트워크를 새로운 체인으로 확장하는 과정을 가속화한다.

3.0.1 LayerZero 패킷 전송

각 구성 요소를 자세히 설명하기 전에, 그림 4에 나타난 것처럼 LayerZero에서 패킷이 전송되는 방식을 개괄한다. LayerZero 메시 네트워크는 프로토콜 관리자가 연결된 각 블록체인에 *LayerZero 엔드포인트(Endpoint)*를 배포함으로써 형성된다. 이 예시에서 OApp은 LayerZero 메시지를 *발신자(sender)* 컨트랙트에서 *수신자(receiver)* 컨트랙트로 LayerZero 메시 네트워크를 통해 전송한다. 설명을 위해 본 예시에서 사용하는 MessageLib는 *Ultra Light Node* [6](#)이다.

초기 설정 단계에서 OApp은 출발지와 목적지 블록체인의 LayerZero *엔드포인트*에 자신의 *보안 스택(Security Stack)*을 구성한다. *보안 스택*에 구성된 MessageLib 버전은 *패킷 버전(packet version)*을 결정한다.

①단계에서 발신자는 *출발지* 체인의 LayerZero *엔드포인트*에서 `lzSend`를 호출하며, 메시지 페이로드와 *경로(path)*를 지정한다. 이 경로는 독립적인 검열 저항 채널에 연결되어 있으며, 발신 애플리케이션 주소, 출발지 엔드포인트 ID, 수신 애플리케이션 주소, 목적지 엔드포인트 ID로 구성된다.

출발지 엔드포인트는 이후 패킷에 대해 간격 없는 단조 증가하는 *논스(nonce)*를 할당한다. 이 *논스*는 경로와 연결(concatenate)되고, 그 결과를 해시하여 패킷의 전역 고유 ID(GUID)를 계산한다. 이 GUID는 오프체인 및 온체인 작업자들(예: 실행자, DVN)이 LayerZero 메시지의 상태를 추적하고 동작을 트리거하는 데 사용된다.

필드명	타입
패킷 버전(Packet version)	uint8
논스(Nonce)	uint64
출발지 엔드포인트 ID	uint32
발신자(Sender)	uint256
목적지 엔드포인트 ID	uint32
수신자(Receiver)	uint256
GUID	uint256
메시지 페이로드	bytes[]

표 2: LayerZero 패킷은 헤더와 본문으로 구성된다. 헤더에는 패킷 버전과 경로가 포함되며, 본문은 실제 메시지 페이로드로 구성된다. 패킷은 전역 고유 식별자(GUID)로 식별된다.

출발지 엔드포인트는 OApp의 *보안 스택(Security Stack)*을 읽어 패킷을 인코딩하기 위한 올바른 안전한 MessageLib (이 예에서는 ULN)를 결정한다. 출발지 MessageLib는 구성된 보안 스택에 따라 패킷을 처리한 후, 구성된 DVN으로 전파하여 목적지 MessageLib에서 메시지를 검증하고, 필요 시 지정된 실행자(executor)에게 전달하여 오프체인 동작을 트리거한다. 이러한 DVN 및 실행자 식별자와 관련 인자들은 MessageLib에 의해 *메시지 옵션(Message Options)*이라 불리는 비구조적 바이트 배열로 직렬화된다. ULN이 패킷을 인코딩하고 엔드포인트에 반환하면, 엔드포인트는 패킷을 내보내며 LayerZero 송신 트랜잭션을 종료한다.

②단계에서 구성된 DVN들은 각각 독립적으로 목적지 MessageLib에서 패킷을 검증한다. ULN의 경우 이는 패킷 페이로드의 해시를 저장하는 것을 의미한다. 일정 임계값 이상의 DVN이 페이로드를 검증하면 (3.2.1절 참조), 작업자(예: 실행자, DVN, 사용자)가 ③단계에서 패킷을 엔드포인트에 커밋한다. 엔드포인트는 무손실 채널에 커밋하기 전에 페이로드 검증이 OApp에 의해 구성된 보안 스택을 반영하는지 확인한다.

마지막으로 ④단계에서 실행자는 커밋된 메시지에 대해 lzReceive를 호출하여 수신자 OApp의 로직을 해당 패킷에 대해 실행한다. 채널이 무손실·정확히 한 번 전달을 보장할 수 없는 경우, ④단계는 검열 방지를 위해 되돌려진다(revert).

3.1 LayerZero 엔드포인트

*LayerZero 엔드포인트(Endpoint)*는 불변의 오픈소스 스마트 컨트랙트로 구현되어 체인마다 하나 이상의 인스턴스로 배포되며, 안정적인 애플리케이션 지향 인터페이스(표 3), 정확히 한 번 전달을 보장하는 무손실 네트워크 채널의 추상화, 그리고 OApp 보안 스택(Security Stack) 관리를 제공한다. LayerZero 엔드포인트의 불변성은 업데이트 격리, 구성 소유권, 채널 무결성을 강제하여 장기적인 채널 유효성을 보장한다. 보안 스택은 OApp과 무허가 DVN 집합 간의 신뢰-비용 관계를 조정하기 때문에 LayerZero의 채널 생존성 보장에 핵심적이다.

OApp은 엔드포인트에서 send를 호출하여 LayerZero를 통해 전송될 메시지를 큐에 넣으며, 이때 경로(표 2), 메시지 페이로드, 그리고 MessageLib에 의해 해석될 직렬화된 옵션을 담은 선택적 바이트 배열(메시지 옵션, Message Options)을 지정한다. 메시지 옵션은 의도적으로 비구조적이며, 4장에서 다루는 사용 사례처럼 확장성을 향상시킨다. send의 대응 함수는 lzReceive로, 이는 목적지 체인에서 지정된 GUID를 가진 메시지를 소비하기 위해 실행된다. 목적지 체인에서 엔드포인트는 lzReceive와 getInboundNonce 호출을 처리하며, 무손실·정확히 한 번 전달을 강제하여 메시지 채널의 무결성을 보호한다. lzReceive는 메시지가 무손실로 전달될 수 있는 경우 해당 메시지의 검증된 페이로드를 OApp에 전달한다. getInboundNonce는 무손실로 전달 가능한 가장 높은 논스를 반환하며, 이는 0부터 해당 논스까지의 모든 메시지가 검증되었거나, 건너뛰었거나, 전달된 가장 높은 논스로 계산된다. 잘못 전송된 메시지나 악의적인 패킷을 처리하기 위해, OApp은 문제가 되는 패킷의 전달을 건너뛰기 위해 clear를 호출하거나, 검증과 전달 모두를 건너뛰기 위해 skip을 호출한다.

`clear`와 `skip` 외에도 엔드포인트는 `nilify`와 `burn`이라는 두 가지 편의 기능을 제공한다. `nilify`는 검증된 패킷을 무효화하여 `MessageLib`에서 새로운 패킷이 커밋될 때까지 해당 패킷이 실행되지 않도록 한다. 이 함수는 손상된 DVN에서 악의적으로 생성된 패킷을 선제적으로 무효화하는 데 사용할 수 있다. `burn`은 OApp이 패킷 내용을 알지 못한 채 `clear`할 수 있도록 하는 편의 함수이다. 이는 잘못된 보안 스택이 잘못된 해시를 엔드포인트에 커밋했거나, OApp이 무효화(`nilify`)된 논스를 정리해야 할 때 유용하다.

3.1.1 순서 무관 무손실 전달

우리는 `LayerZero` 메시지 채널에 대해 협상 불가능한 두 가지 일관성 요구사항을 제시한다: *무손실 전달*과 *정확히 한 번 전달*이다. 검열 저항 채널은 무손실이어야 하며, 재생 공격을 방지하기 위해 정확히 한 번 전달이 요구된다. 이 두 가지 요구사항은 네트워크 무결성에 필수적이며, 기반 블록체인이 결함이 없는 한 프로토콜에 의해 보장된다.

루틴(Routine)	인자 (Arguments)	설명(Description)
send	path	LayerZero를 통해 메시지를 전송한다. 메시지를 전송할 채널의 경로.
	payload	수신자에게 전송할 데이터.
	Message Options	MessageLib이 해석할 인자들의 바이트 배열 (선택적).
getInboundNonce	path	모든 이전 메시지가 수신된 가장 큰 논스를 반환한다. 메시지 채널에 해당하는 경로.
skip	path, nonce	수신자가 특정 논스의 검증 및 전달을 건너뛸 때 호출한다. 건너뛸 메시지의 논스(반드시 수신 논스 + 1).
clear	path, guid, message	수신자가 검증된 논스를 건너뛰기 위해 호출한다. 건너뛸 논스가 있는 경로. 건너뛸 논스의 GUID. 건너뛸 메시지의 내용.
lzReceive	path, nonce, GUID, message, extraData	실행자가 채널에서 메시지를 수신하기 위해 호출한다. 메시지를 수신할 채널의 경로. 수신된 메시지의 논스 출력 파라미터. 수신된 메시지의 GUID. 수신할 메시지. 수신자가 요청한 추가 데이터.
nilify	path, nonce, payloadHash	실행자가 채널에서 메시지를 무효화(nilify)하기 위해 호출한다. 무효화할 메시지가 있는 채널의 경로. 무효화할 패킷의 논스. 무효화할 페이로드의 해시.
burn	path, nonce, payloadHash	실행자가 채널에서 메시지를 소각(burn)하기 위해 호출한다. 소각할 메시지가 있는 채널의 경로. 소각할 패킷의 논스. 소각할 페이로드의 해시.

표 3: LayerZero 핵심 메시징 API.

LayerZero의 채널은 반드시 경로에 의해 분리되고 격리된다(표 2). 그렇지 않으면 두 개의 서로 다른 OApp 간에 공유되는 손실이 채널 유효성이나 생존성을 희생해야 하기 때문이다. 공격적인 애플리케이션은 단순히 패킷 검증을 거부함으로써 공유 채널의 생존성을 쉽게 차단할 수 있으며, 반대로 다른 애플리케이션이 악의적인 OApp의 패킷을 강제로 건너뛰게 허용하는 것은 검열에 해당한다. 각 채널은 간격 없는(strictly gapless), 엄격히 단조 증가하는 양의 정수 논스로 구현된 논리적 시계를 유지하며, 채널을 통해 전송되는 각 메시지는 정확히 하나의 논스를 할당받는다. 목적지 엔드포인트에서는 각 논스가 정확히 하나의 검증된 페이로드 해시에 매핑되며(3.2.2절), 채널은 각 전달된 페이로드가 해당 논스의 검증된 해시에 대응되도록 강제한다. LayerZero는 특정 패킷의 전달이 동일 채널의 더 낮은 논스를 가진 모든 패킷이 이미 전달되었거나,

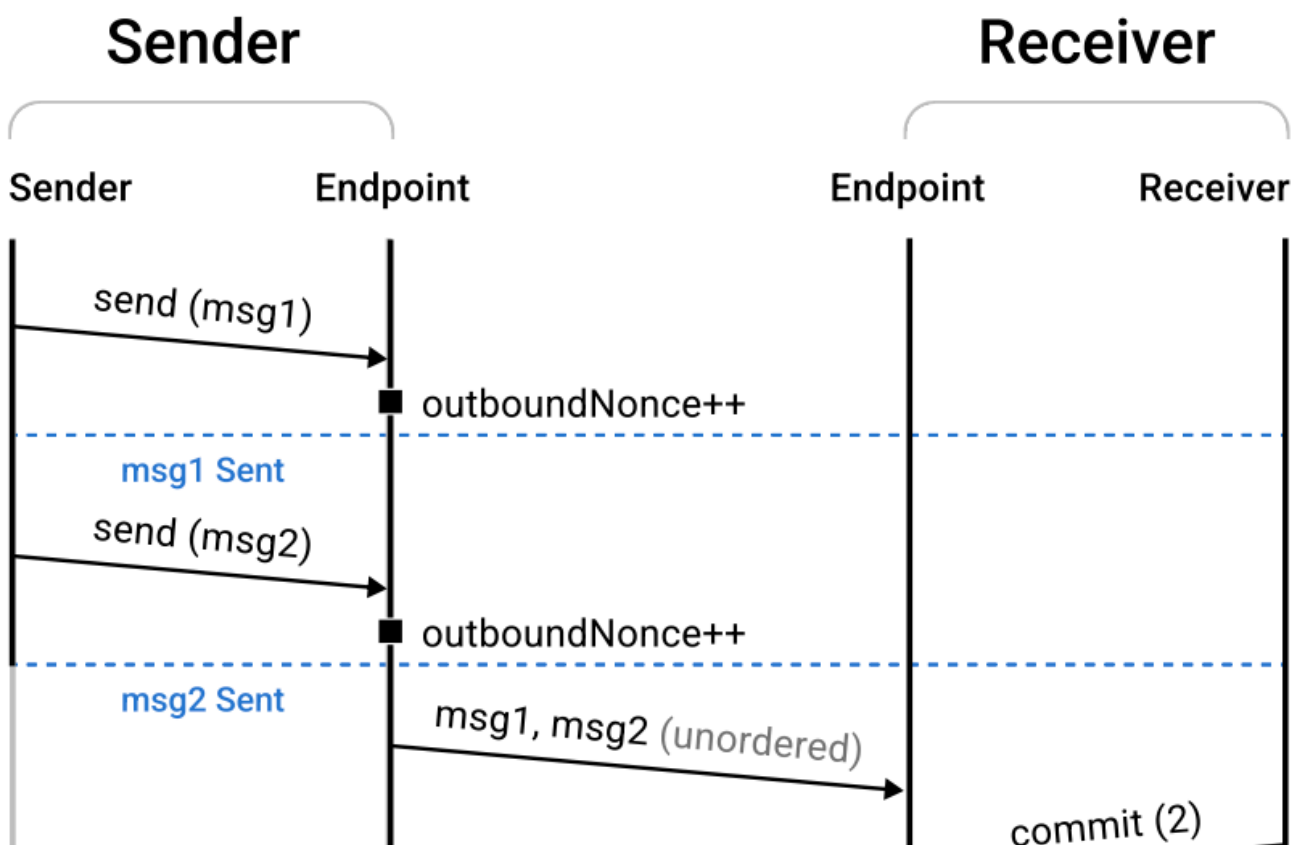
전달 가능하거나, 혹은 건너뛰어졌음을 보장한다.

주어진 채널에서 두 메시지 m_k 와 m_{k+1} 이 각각 양의 정수 논스 k 와 $k + 1$ 을 할당받았다고 가정하자. 간격 없는 논스는 $(\exists m_j \text{ s.t. } m_k \rightarrow m_j \wedge m_j \rightarrow m_{k+1})$ 을 보장하며, 여기서 $\rightarrow$ 는 happens-before 관계를 의미한다. 비공식적으로 설명하면, k 와 $k + 1$ 사이에 논스를 가진 패킷은 존재할 수 없다는 뜻이다. 이는 무손실성을 보장하는 가장 약하면서도, 그만큼 가장 유연한 조건이다. 더 강력한 조건(예: 엄격한 순차적 전달)은 필요하다면 이 추상화 위에 추가적으로 부과될 수 있다.

검열 저항성은 모든 이전 논스가 커밋되거나 건너뛰어지지 않으면 어떠한 논스도 전달될 수 없도록 강제함으로써 구현된다. 예를 들어, 논스 N 을 가진 패킷은 $1, \dots, N-1$ 의 논스를 가진 모든 패킷이 커밋되거나 수신자에 의해 명시적으로 건너뛰어진 경우에만 전달될 수 있다. 우리는 실행될 수 있는 가장 큰 논스를 *인바운드 논스(inbound nonce)*라 부른다.

무손실성과 정확히 한 번 전달은 Zarick 등 [6]이 입증한 바와 같이, 엄격한 순차적 검증 및 실행을 통해 달성할 수 있다. 그러나 전달 순서를 강제하면 특정 블록체인에서 인위적인 처리량 한계가 발생하고 오프체인 인프라를 복잡하게 만든다. LayerZero에서는 이러한 순서 제약을 완화하여, 채널 무결성을 유지하면서도 추가적인 온체인 계산 오버헤드를 발생시키지 않는 순서 무관(out-of-order) 전달을 구현한다.

무손실·정확히 한 번·순서 무관 전달을 보장하는 검열 불가능 채널의 유일하게 효율적인 온체인 구현은 가장 높은 전달된(delivered) 논스를 추적하는 것이며, 이를 우리는 *게으른 인바운드 논스(lazy inbound nonce)*라 부른다. 게으른 인바운드 논스는 0에서 시작하며, 패킷은 게으른 인바운드 논스부터 해당 패킷 논스까지 모든 패킷이 검증된 경우에만 실행될 수 있다. 패킷이 전달되거나 건너뛰어질 때마다, 게으른 인바운드 논스는 현재 값과 전달된 패킷의 논스 중 더 큰 값으로 갱신된다.



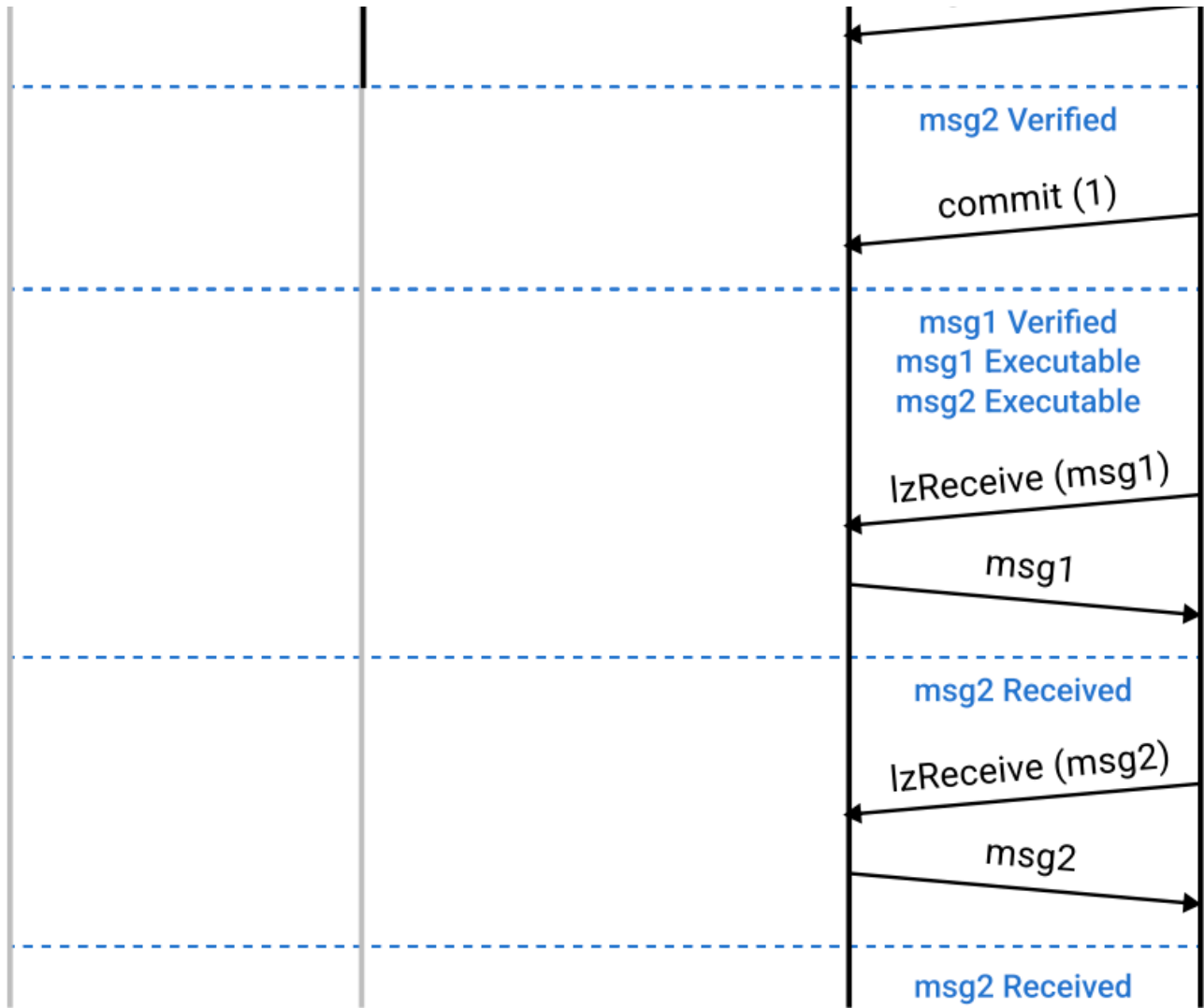


Figure 5: A packet is *Sent* after source transaction increments the nonce, *Verified* after it is committed into the Endpoint, and *Received* after delivery (execution).

효율적인 무손실·정확히 한 번 전달을 달성할 수 있는 또 다른 이론적 알고리즘은 *실행*이 아닌 *검증* 시점에 게으른 인바운드 논스를 갱신하는 것이다. 그러나 이는 실제로 작동하지 않는다. 단일 패킷 커밋이 게으른 인바운드 논스에서 임의로 큰 폭의 업데이트를 초래할 수 있기 때문이다. 예를 들어, 논스 2부터 1000까지가 논스 1보다 먼저 커밋된 경우, 논스 1을 커밋하려면 게으른 인바운드 논스를 갱신하기 위해 1000개의 논스를 모두 순회해야 한다. 이는 반복 횟수가 기반 블록체인의 한계를 초과하면 해당 채널이 영구적으로 생존성을 상실하는 상황을 만든다.

반면, 실행 시점에서 게으른 인바운드 논스를 갱신하는 경우 계산 한계에 부딪힐 수는 있지만, 더 낮은 논스에서 무허가로 실행을 재시도하면 성공할 수 있다. 실행 시 항상 되돌려지는 메시지(즉, 영구적으로 실행 시 되돌려질 메시지)들의 연속적인 시퀀스가 블록체인 반복 한도보다 길 경우, 채널 생존성이 일시적으로 손실될 수 있다. 그러나 이 상황은 OApp 소유자가 `clear`(표 3)를 호출하여 이러한 실행 불가능한 패킷들의 실행을 건너뛰으로써 쉽게 해결할 수 있다.

정확히 한 번 전달을 강제하기 위해, 우리는 패킷이 성공적으로 전달된 후 해당 패킷을 플래그한다. *LayerZero*에서는 이는 전달된 패킷의 검증 해시를 무손실 채널에서 삭제하고, 게으른 인바운드 논스 이하의 논스에 대한 검증을 허용하지 않음으로써 구현된다.

그림 5는 *LayerZero* 패킷의 수명 주기를 단순화된 관점에서 보여주며, 무손실 채널을 설명한다. 이 예에서 *OApp*은 출발지 체인에서 목적지 체인으로 두 개의 메시지를 비동기적으로 전송하며, 각 패킷은 *Sent*, *Verified*, *Received*의 세 가지 상태 중 하나에 있을 수 있다. 패킷은 출발지 엔드포인트가 패킷에 논스를 할당한 후 *Sent* 상태가 되며, 이후 요청된 DVN들이 목적지 *MessageLib*에서 이를 검증한다. 실행자가 *commitVerification*(그림 5에서는 *commit*으로 표시됨)을 호출하여 해당 패킷이 *OApp*의 보안 스택에 의해 검증되었음을 확인하면 패킷은 *Verified* 상태로 전환된다.

패킷과 그 이전의 모든 패킷이 커밋되면, 실행자가 *lzReceive*를 호출하여 패킷을 전달하고, 트랜잭션이 되돌려지지 않는 한 패킷은 *Received* 상태로 전환된다. 하나 이상의 이전 논스가 커밋되지 않은 경우, 무손실 채널은 검열을 방지하기 위해 되돌려진다.

3.2 MessageLib

*MessageLib 레지스트리*는 *MessageLibs*의 모음으로, 각 라이브러리는 출발지 체인에서 패킷을 안전하게 내보내고 목적지 *MessageLib*에서 이를 검증하는 역할을 담당한다. 각 독립적인 *MessageLib*는 외재적 보안을 구현하며, 이는 기반 환경 변화에 대한 적응을 필요로 하고, *MessageLib 레지스트리*의 완전 불변 설계를 배제한다. *MessageLib*는 각 패킷의 *페이로드 해시*를 검증하고, 외재적 보안 요건(예: DVN 임계값)이 충족되면 엔드포인트에 검증된 페이로드 해시를 커밋한다.

외재적 보안의 확장성을 제공하면서도 기존 *OApp*이 현장 업데이트(*in-place updates*)로부터 보호되도록 하기 위해, 우리는 *MessageLib 레지스트리*를 불변 라이브러리의 추가 전용(*append-only*) 레지스트리로 구성하였다. 각 라이브러리는 프로토콜 인터페이스를 준수하는 한 임의의 검증 메커니즘을 구현할 수 있다. 이러한 설계는 대부분의 메시징 서비스가 빠지는 검증 종속(*validation lock-in*)의 함정을 피하며, *LayerZero*가 모든 상황에서 가장 안전하고 효율적인 검증 알고리즘을 활용할 수 있도록 보장한다.

이 설계는 처음에는 직관에 반하는 것처럼 보일 수 있다. 이는 어떤 현장 소프트웨어 업데이트도 허용하지 않으므로 프로토콜 관리자가 소프트웨어 버그를 해결하지 못하게 만드는 것처럼 보이기 때문이다. 그러나 단일 주체에게 현장에서 문제를 일방적으로 수정할 권한을 부여하는 것은 동시에 새로운 취약점을 도입할 권한을 주는 것이기도 하다. 이는 악의적인 코드 변경이 장기적 프로토콜 보안 불변성을 쉽게 위반할 수 있게 만들며, 그 결과 보안 불변성을 무효화한다. *MessageLib*에서 불변 라이브러리를 채택하기로 한 우리의 결정은 새로운 코드와 버그 수정의 진입 장벽을 높이는 결과를 낳지만, 이는 *LayerZero*에서 본질적 보안을 실현하는 데 필수적이다.

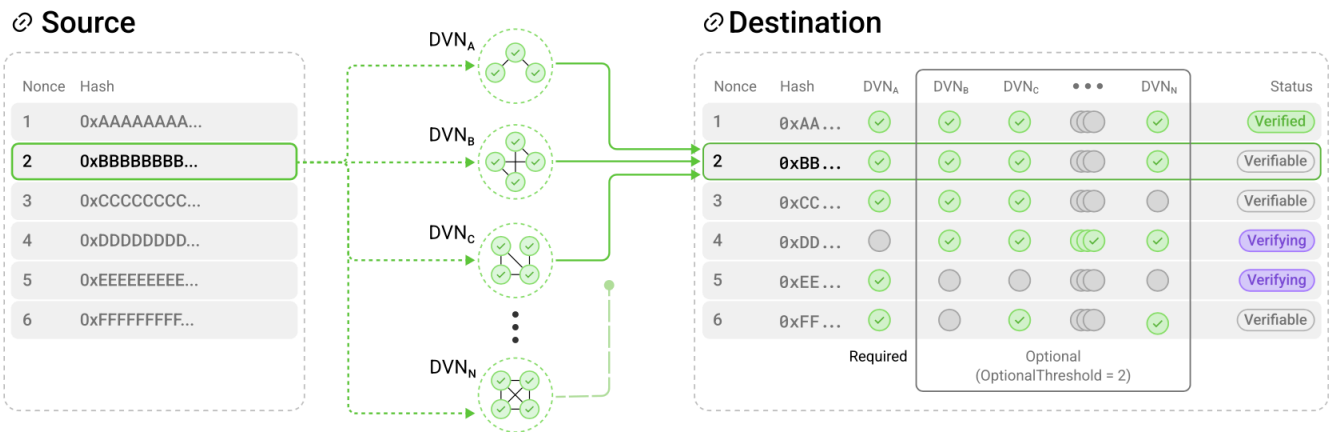


Figure 6: The Ultra Light Node enforces (onchain) the configured required DVNs, optional DVNs, and OptionalThreshold. Verification is neither lossless nor ordered, and messages can be committed to the channel as soon as the Security Stack is fulfilled.

우리는 이 추가 전용(append-only) 설계가 확장성을 훼손하지 않으면서 본질적 보안을 구현할 수 있는 유일한 방법이라고 주장한다. 이론적으로 완전히 버그가 없고 완벽하게 최적화된 단일 비업그레이드형 *MessageLib*는 본질적인 종단 간 보안을 제공할 수 있지만, 이는 비현실적인 상황에서도 실질적으로 불가능하다. 실행 환경(예: opcode의 제거 또는 추가), 합의 메커니즘(예: 검증인 선출 프로토콜), 검증 알고리즘에 대한 애플리케이션의 진화하는 선호도 등은 필연적으로 프로토콜 업데이트를 필요로 한다.

*MessageLib*의 현장 수정(in-place modifications)을 허용하는 어떤 방식도 본질적으로 안전하지 않다. OApps는 프로토콜 관리자의 개발자, 감사자, 거버넌스 구조에 의존하게 되며, 업데이트된 코드의 보안성이 기존 *MessageLib*과 동일하거나 더 뛰어나다는 것을 보장할 실질적인 방법이 없다. OMP는 확장을 허용해야 하지만, 동시에 이전 버전의 외재적 보안이 이러한 코드 추가에 의해 결코 영향을 받지 않도록 보장해야 한다. 따라서 본질적으로 안전한 업데이트를 제공하는 유일한 설계는 새로운 코드베이스 버전을 불변의 라이브러리 버전 레지스트리에 추가하는 것이다.

각 *MessageLib*는 독립적으로 작동하며 다음 작업을 처리한다: (1) *엔드포인트*로부터 메시지를 수락, (2) 패킷(표 2)을 인코딩 및 전송하여 DVN과 실행자에게 전달하고 필요한 수수료 지불, (3) 목적지 체인에서 패킷 검증, (4) 검증된 메시지를 목적지 *엔드포인트*에 커밋. 나머지 모든 작업은 실행자에 의해 처리되므로 *MessageLib*의 코드 크기를 최소화하고, 새로운 실행자의 구현과 운영을 통해 기능을 쉽게 추가할 수 있다. LayerZero 관리자(admin)를 포함한 어떤 주체도 일단 *MessageLib* 레지스트리에 추가된 라이브러리를 수정하거나 제거할 수 없다.

무손실성은 *MessageLib* (검증 계층)가 아니라 불변의 엔드포인트(실행 계층)에서 보장된다는 점에 주목해야 한다. 3.2.1절에서 설명하듯이, *MessageLib*는 검증된 패킷을 순서가 어긋나거나 간격이 있는 상태로 인바운드 큐에 기록할 수 없다. 그러나 검증된 패킷들의 시퀀스에 간격이 존재한다면, 무손실 채널에서 패킷을 소비할 수 없다.

3.2.1 Ultra Light Node

Ultra Light Node (ULN)은 모든 LayerZero 배포에 포함되는 기본 MessageLib이며, 사용자 정의 가능한 2계층 합의(quorum) 의미론을 통해 최대 254개의 DVN을 조합할 수 있도록 한다. ULN은 어떤 검증 알고리즘에도 필요한 최소한의 근본적 기능 집합을 구현하므로 모든 블록체인과 보편적으로 호환된다. ULN을 사용하도록 구성된 각 OApp *보안 스택(Security Stack)*에는 필수(required) DVN 집합(X), 선택적(optional) DVN 집합, 그리고 임계값(OptionalThreshold)이 포함된다. X DVN은 필수이며, 해당 패킷은 모든 X 필수 DVN과 최소한 OptionalThreshold 개수의 선택적 DVN이 해당 페이로드 해시에 서명해야만 전달될 수 있다. 필요한 DVN 서명이 ULN에 집계된 후, 해당 패킷은 엔드포인트에 커밋될 수 있다. 필수 DVN 모델은 OApp이 검증 계층의 외재적 보안에 대해 하한선을 설정할 수 있게 하며, 필수 집합 내에서 가장 안전한 DVN의 서명 없이는 어떤 메시지도 검증될 수 없도록 한다. 이러한 설계는 외재적 보안의 대부분을 DVN에 위임하면서도 여전히 보안 스택을 온체인에서 강제한다.

우리는 그림 6에서 Ultra Light Node 검증 의미론의 예시를 설명한다. OApp 보안 스택에는 필수 DVN(DVN_A)과 $N - 1$ 개의 선택적 DVN(DVN_B , DVN_C , $\dots$)이 포함되어 있으며, OptionalThreshold는 1이다. 이는 DVN_A 에 “거부권(veto)”을 부여하며, 패킷이 커밋되기 위해서는 최소한 하나의 선택적 DVN이 패킷을 검증해야 한다. 논스 1은 메시징 채널에 커밋되었으며(Verified), 논스 2, 3, 6은 보안 스택 요건이 충족되어 커밋 가능(verifiable)하지만, 실행자가 commitVerification을 호출하기 전까지는 커밋되지 않는다. 반면 논스 4와 5는 각각 필수 검증자 집합 및 OptionalThreshold 요건을 충족하지 않았으므로 커밋할 수 없다.

이와 같은 조합 가능한 검증 원시(primitive)는 OApp이 비용과 보안 간의 균형을 조정할 수 있도록 하며, DVN 집합 내에서 손쉽게 클라이언트 다양성을 구성할 수 있게 하고, 외재적 보안 업그레이드의 엔지니어링 비용(온체인 코드 확장 없음)을 최소화한다. 클라이언트 다양성의 중요성은 아무리 강조해도 지나치지 않으며, 단 하나의 DVN이 손상되더라도 버그가 있는 코드에 노출될 수 있기 때문이다.

3.2.2 MessageLib 버전 관리 및 마이그레이션

MessageLibs는 고유한 ID와 의미적(major.minor) 버전으로 식별되며, 두 엔드포인트가 동일한 major 버전을 가진 MessageLib을 구현한 경우에만 메시지를 전송할 수 있다. Major 버전은 패킷 직렬화 및 역직렬화의 호환성을 결정하며, minor 버전은 버그 수정 및 기타 비파괴적(non-breaking) 변경을 위해 예약된다. 각 LayerZero 메시지의 패킷 버전은 MessageLib 버전에 매핑되며, DVN은 이를 사용하여 목적지 블록체인에서 패킷 검증을 제출할 MessageLib을 식별한다. 각 OApp *보안 스택(Security Stack)*은 자신이 걸쳐 있는 각 체인에 대해 사용할 sendLibrary와 receiveLibrary를 지정한다. sendLibrary는 메시지를 전송할 때 사용할 라이브러리의 ID 및 버전이며, receiveLibrary는 메시지를 수신할 때 사용할 라이브러리의 ID 및 버전이다. 이러한 구성 가능성은 OApp이 개별 요구사항에 따라 보안 스택의 비용과 보안을 조정할 수 있게 한다.

빠른 프로토타이핑을 위해 LayerZero는 OApp이 자신의 보안 스택을 LayerZero 관리자가 선택·유지하는 기본값으로 지연(lazy) 해결할 수 있도록 하는 옵트인(opt-in) 메커니즘을 구현한다. 그러나 OApp 소유자는 실제 운영 애플리케이션에서는 반드시 자신의 보안 스택을 명시적으로 설정하는 것이 권장된다. OApp이 해당 경로에 대해 자신의 보안 스택을 설정하거나 기본 보안 스택에 옵트인하기 전까지는 메시지를 수신할 수 없다.

비동기 네트워크에서 원자적 트랜잭션 조정이 불가능하다는 점[2, 5]은 OApp 보안 스택을 재구성할 때 라이브 마이그레이션 프로토콜을 필요로 한다.

동일한 major 버전 내에서 MessageLib 업그레이드(예: 1.1 → 1.2)는 단순히 sendLibrary 및/또는 receiveLibrary를 원하는 버전으로 설정함으로써 이루어진다. 그러나 서로 다른 major 버전 간의 마이그레이션(예: 1.2 → 2.0)은 더 복잡하다. 먼저 OApp은 이전 receiveLibrary(1.2)에 대해 유예 기간(grace period)을 설정하며, 이 기간 동안 OApp이 sendLibrary를 2.0으로 설정하더라도 receiveLibrary 1.2는 계속해서 메시지를 수신할 수 있다. 다음으로, OApp은 sendLibrary를 새로운 버전(2.0)으로 설정한다. 이후 모든 신규 메시지는 새로운 패킷 버전을 가지지만, 유예 기간이 끝날 때까지는 이전 receiveLibrary 1.2와 2.0 모두 메시지를 검증할 수 있다. 유예 기간이 만료되면 오직 2.0만 메시지 검증이 허용된다. 만약 유예 기간이 끝나기 전에 모든 진행 중인 LayerZero 메시지가 목적지 엔드포인트에 커밋되지 못하면, 패킷 전달이 일시적으로 중단되며 OApp은 모든 진행 중인 메시지를 커밋하기 위해 receiveLibrary를 이전 버전(1.2)으로 다시 재구성해야 한다. 이러한 방식으로 LayerZero는 본질적으로 안전하면서도 마찰 없는 라이브 마이그레이션을 MessageLib 버전 간에 구현한다.

3.3 탈중앙화 검증자 네트워크 (Decentralized Verifier Network)

LayerZero의 데이터링크는 동기성 가정을 두지 않고 두 블록체인을 연결하려면 하나 이상의 제3자를 통한 통신이 필요하다는 근본적 관찰에 기반해 설계되었다[5]. DVN이 잠재적으로 오프체인 기반일 수 있다는 점의 결과는 장기적인 불변성과 가용성을 보장할 수 없다는 것이며, 따라서 허가형 검증 모델은 채널 생존성에 대한 강력한 보장을 본질적으로 제공할 수 없다. 이에 따라 LayerZero는 누구나 운영할 수 있고, 무허가로 자신만의 DVN을 LayerZero에 통합할 수 있는 무허가형(permissionless), 구성 가능한 검증 모델을 구현하였다. 이름에서 알 수 있듯이, *탈중앙화 검증자 네트워크(DVN)*는 내부적으로 검증자 집합으로 구성되어 있으며, 이들이 분산 합의를 통해 출발지 블록체인에서 패킷 해시를 안전하고 신뢰성 있게 읽는다. 이 설계는 단일 DVN 내부에서도 클라이언트 다양성을 허용하여, 단일 검증자의 결함으로 인한 프로토콜 중단이나 오류 가능성을 최소화한다.

이 모델은 다른 메시징 서비스의 두 가지 뚜렷한 결함, 즉 공유 보안(shared security)과 유한 내결함성(finite fault tolerance)을 극복한다. 기존 크로스체인 메시징 서비스는 모든 클라이언트가 공유하는 단일 보안 구성을 제공하는데, 이는 두 가지 측면에서 비효율적이다. 첫째, 프로토콜이 지정한 검증자 집합이 손상되거나 결함이 발생했을 경우 OApp이 취할 수 있는 조치가 없다. 둘째, 대부분의 OApp이 당면한 작업을 위해 불필요하게 비용이 많이 들거나 지나치게 위험한 보안 구성 중 하나를 선택해야만 한다. 유한 내결함성 문제는 모든 허가형 검증 모델 기반 메시징 서비스가 겪는 문제이며, 제한된 수의 허가형 운영 조직 집합이 실패하면 전체 프로토콜이 영구적이고 회복 불가능한 실패에 직면하게 된다.

유형(Type)	구조(Structure)
실행 가스	[TYPE.1, executionGas]
가스 및 네이티브 드롭	[TYPE.2, executionGas, natedropAmount, receiverAddress]
복합(Composite)	[TYPE.3, {workerID, opType, length, command}, ...]

표 4: 메시지 옵션(Message Options)은 옵션 유형을 식별하기 위한 매직 넘버로 시작하며, 이후 유형별 옵션이 이어진다. 유형 1과 2는 각각 실행 가스 한도를 설정하고, 옴니체인 트랜잭션의 일부로 추가 네이티브 가스 토큰을 전송하는 데 특화되어 있다. 유형 3은 임의의 오프체인 작업자 집합에 대한 인자를 내포한다.

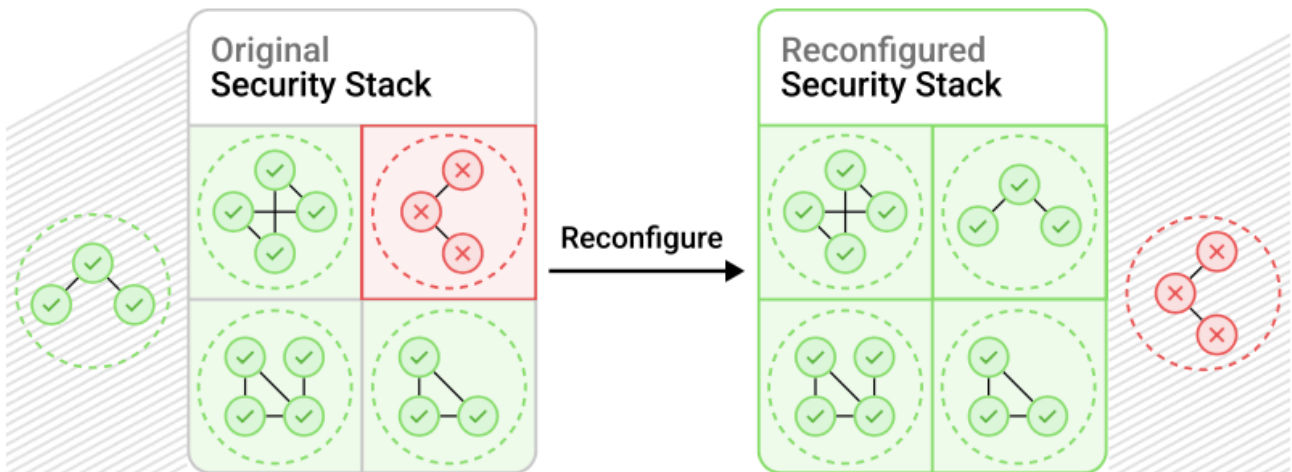


Figure 7: OApps can easily reconfigure their Security Stack to exclude faulty DVNs.

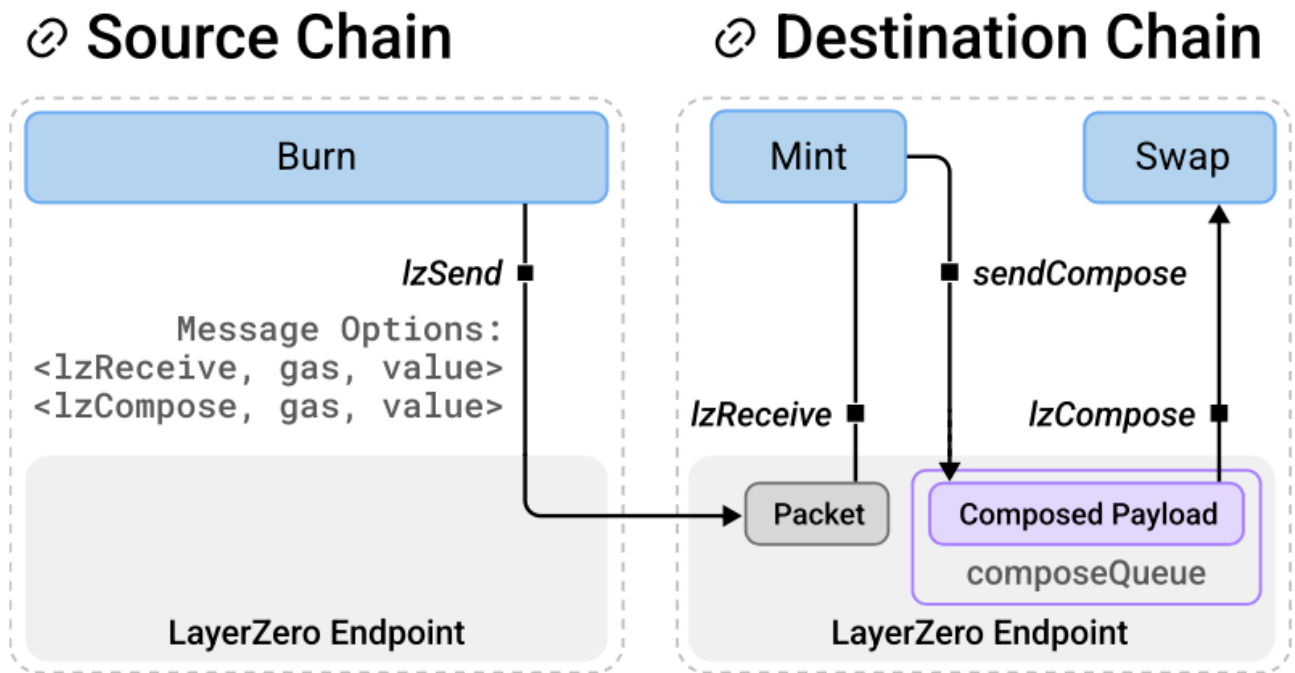


Figure 8: lzCompose enables chain-agnostic composition with liveness and safety closures.

무허가 방식으로 DVN이 운영되기 때문에, *LayerZero*는 사실상 무제한 수준의 내결함성을 제공할 수 있다. 설령 기존 DVN 모두가 소프트웨어 버그, 보안 침해, 자연 재해, 운영/거버넌스 문제로 인해 생존성을 상실 하더라도, OApp 개발자는 독자적으로 DVN을 운영하여 프로토콜의 지속적 운영을 보장할 수 있다. 또한 OApp은 *보안 스택* 재구성을 통해 트래픽을 원활하게 재라우팅할 수 있으므로, 손상된 오프체인 인프라로부터의 복구가 가능하다(그림 7).

3.4 실행자 (Executors)

외재적으로 안전한 코드를 구현하고 갱신하는 것은 엄격한 보안 테스트와 감사 요건 때문에 자원 집약적이다. 이러한 엔지니어링 과제는 다양한 옴니체인 애플리케이션의 요구를 지원할 수 있도록 *LayerZero*를 쉽게 확장 가능하게 만들려는 우리의 목표와 충돌한다. *LayerZero*는 검증과 실행을 분리함으로써 이 문제를 해결한다. 보안에 핵심적이지 않은 코드는 *실행자(executors)*로 분리되며, 이들은 무허가(permissionless)이며 패킷 검증 범위와 격리되어 있다. *MessageLib*과 실행자 간의 보안 핵심 코드와 “기능” 코드의 분리는 두 가지 주요 이점을 제공한다. 첫째, 애플리케이션이 보안을 고려하지 않고도 기능 확장을 사용, 구현, 조합할 수 있도록 한다. *엔드포인트*는 실행자가 검증되지 않았거나 무손실이 아닌 메시지를 전달하는 것을 방지하므로, 패킷 실행은 검증과 완전히 격리된다. 둘째, *LayerZero*에서 보안과 생존성이 분리되어, 결함 있는 실행자가 메시지 전달을 일방적으로 차단할 수 없도록 보장한다. 이러한 방식으로 검증 계층과 실행 계층을 분리함으로써, 애플리케이션은 오류가 어느 계층에서 발생했는지를 손쉽게 디버깅할 수 있다.

OApp이 *LayerZero* 메시지를 전송할 때, 모든 오프체인 작업자(예: 실행자, DVN 등)와 해당 인자들을 *MessageLib*이 해석하는 바이트 배열인 *메시지 옵션(Message Options)*을 통해 지정한다. 이후 실행자는 *보안 스택(Security Stack)*이 패킷을 검증할 때까지 대기하며, 검증이 완료되면 *메시지 옵션*에 인코딩된 명령에 따라 동작을 수행한다.

우리는 *메시지 옵션*을 의도적으로 비구조적으로 남겨두었다(4절 참조). 더 제한적인 API는 미래 온체인 및 오프체인 작업자의 잠재적 역량을 제한할 뿐이기 때문이다. 프로토콜 수준에서 우리의 주요 목표는 현재 애플리케이션의 필요에 특화된 인터페이스를 만드는 것이 아니라, 가능한 가장 높은 수준의 확장성을 제공하는 것이다.

실행자가 검증 관련 코드와 격리됨으로써 채널 유효성이 간접적으로 향상되며, 무허가 실행은 채널 생존성을 직접적으로 개선한다. 이러한 설계는 *MessageLib*의 코드 풋프린트를 줄이고, 나아가 잠재적 취약점을 최소화한다.

메시지가 *보안 스택*에 의해 검증되면, 가스 비용을 지불할 의사가 있는 누구나 무허가 방식으로 메시지를 실행할 수 있다. 이는 이론적으로 최종 사용자조차 실행자 실패 이후 수동으로 OApp 복구를 트리거할 수 있게 한다.

4 확장 기능

이 절에서는 추가 실행 기능을 통해 프로토콜이 어떻게 확장될 수 있는지 몇 가지 예시를 들어 *LayerZero*의 유연성을 설명한다.

4.1 메시지 옵션 (Message Options)

*메시지 옵션(Message Options)*에 인자를 직렬화하는 단일 표준 형식은 존재하지 않지만, 개발자가 모든 *MessageLib*마다 *메시지 옵션*을 지원하기 위해 맞춤 코드를 작성해야 한다고는 기대하지 않는다. 이를 해결하기 위해, *LayerZero*는 현재 라이브러리 버전 간의 하위 호환성을 용이하게 하기 위해 세 가지 표준화된 *메시지 옵션* 형식을 정의한다(표 4 참조).

유형 1과 2는 단일 실행자가 일반적으로 요구되는 기능을 실행하기 위한 인자를 지정하며, 유형 3은 (workerID, option) 튜플 목록을 인코딩하여 임의 개수의 인자를 임의 개수의 작업자에게 전달할 수 있도록 한다. 실행자에 의해 전달되는 모든 메시지는 이미 검증 계층에 의해 검증되었으므로, 실행자는 메시지 무결성을 훼손하지 않고도 임의의 동작을 수행할 수 있다.

4.2 의미론적으로 통일된 합성

*LayerZero*는 크로스체인 합성을 위해 보편적으로 표준화된 인터페이스 *lzCompose*를 정의한다. 목적지 체인의 전달 트랜잭션을 다른 컨트랙트와 합성하는 것은 임의의 런타임 디스패치를 네이티브하게 지원하는 EVM에 익숙한 사람들에게는 사소해 보일 수 있다. 그러나 기존의 MoveVM 기반 블록체인의조차도 이 기능을 네이티브하게 지원하지 않으므로, EVM 스타일 런타임 디스패치 합성 의미론의 보편성이 무효화된다.

컨트랙트를 합성할 때, 수신자는 먼저 엔드포인트에 합성된 페이로드를 저장(`sendCompose`)하고, 이후 이를 원장으로부터 가져와 `lzCompose`를 호출하여 합성된 콜백으로 전달한다. 이러한 설계는 표면적으로는 EVM 기반 체인에서 비효율적으로 보일 수 있으나, 모든 블록체인에서 합성 의미론을 통일한다.

`lzCompose`는 *LayerZero* 메시징의 정확히 한 번, 무손실 실행 의미론을 그대로 상속받는 의미론적으로 보편적인 표준 합성 원시(`primitive`)를 제공하며, `OApp`이 단일 실행 아키텍처를 정의해 모든 기존 및 미래 블록체인으로 보편적으로 확장될 수 있도록 한다. 그림 8은 하나의 *LayerZero* 트랜잭션에서 토큰을 브리징하고 스왑하는 데 `lzCompose`를 사용하는 예를 보여준다.

`lzCompose` 원시는 데이터 유효성과 채널 생존성을 위한 클로저(`closure`)를 정의하는 강력한 도구로, 합성된 각 컨트랙트를 다른 컨트랙트에 의해 발생할 수 있는 무결성 침해로부터 격리한다. 일단 데이터가 `lzCompose`를 통해 성공적으로 처리되면, 생존성과 무결성은 “닫힌” 상태가 되며, 합성된 컨트랙트에서 발생하는 오류는 그 클로저 *내부에서만* 데이터 생존성이나 데이터 유효성에 영향을 미친다. 합성된 컨트랙트 오류를 클로저 범위에 국한시키는 것은 재진입(`reentrancy`)과 같은 잠재적 공격 표면에 대한 분석을 크게 단순화한다.

`lzCompose`의 또 다른 장점은 복잡한 멀티홉 옴니체인 트랜잭션의 잠재적으로 깊은 호출 스택을 추적하고 분석할 수 있는 통일된 인터페이스를 제공한다는 것이다. 이를 통해 `OApp` 개발자는 옴니체인 코드를 디버깅하는 어려운 작업에 맞설 수 있는 강력한 도구를 얻게 된다.

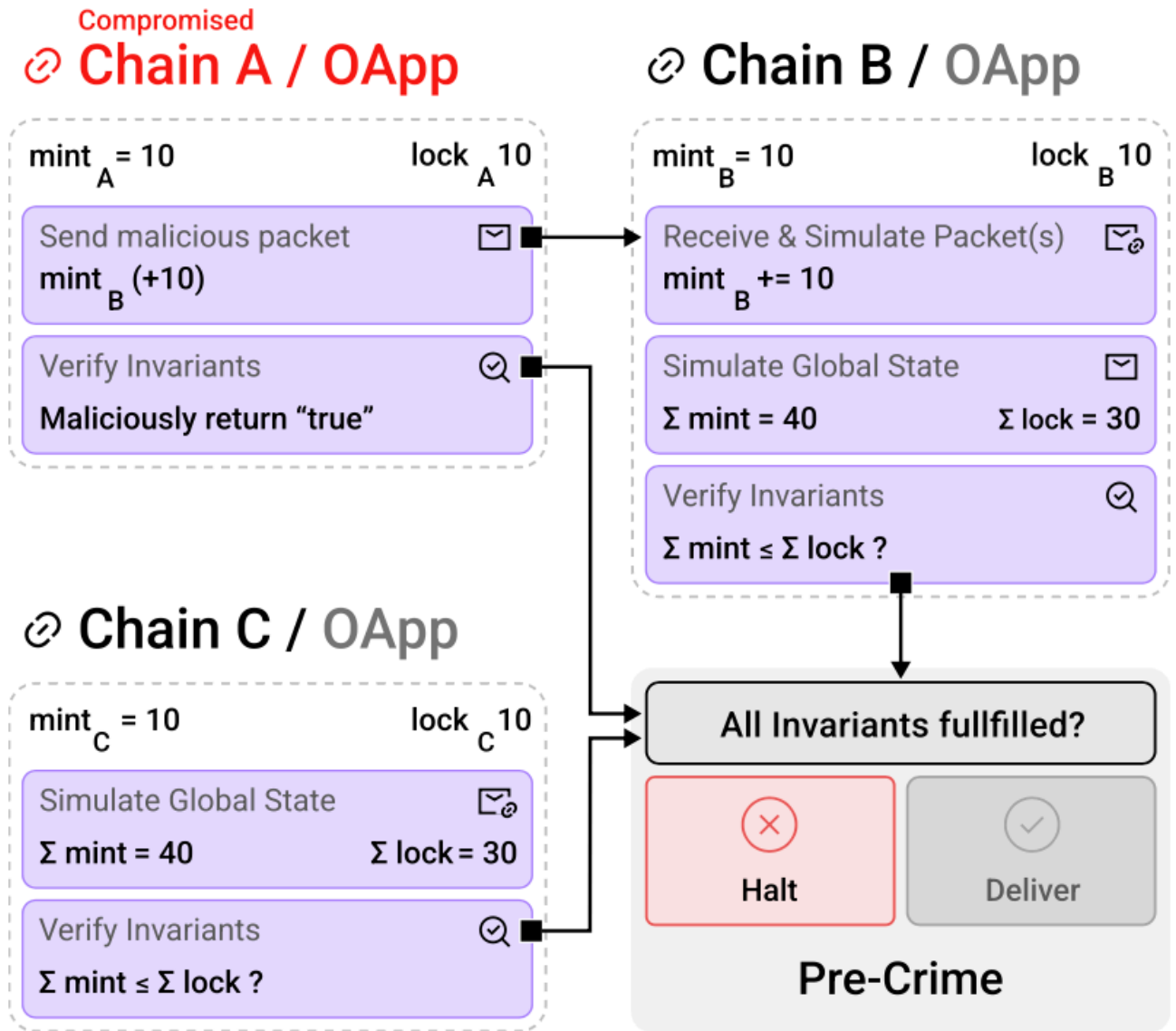


Figure 9: Pre-Crime rejects malicious and malformed messages by checking OApp-specified invariants.

4.3 애플리케이션 수준 보안

메시지 옵션(Message Options) 인터페이스를 사용하여 검증 범위를 확장해 추가 데이터를 포함시키는 것은 불가능하지만, OApp은 이를 활용하여 검증되었지만 악의적인 메시지(예: OApp 수준의 오류를 일으킬 수 있는 버그 메시지)를 감지하고 필터링할 수 있다. 우리는 새로운 오프체인 애플리케이션 수준 보안 메커니즘인 *Pre-Crime*을 소개한다. 이는 기존 *LayerZero* 프로토콜 위에 애플리케이션 특화 패킷 필터링 계층을 추가로 제공한다. *Pre-Crime*은 피어 집합(즉, OApp의 일부 혹은 모든 컨트랙트)이 패킷 전달 결과를 시뮬레이션한 뒤 애플리케이션 보안 불변 조건을 강제할 수 있도록 한다. 불변 조건 검증 결과는 오프체인 작업자에 의해 취합되며, 하나라도 불변 조건 위반을 보고하는 피어가 있다면 해당 패킷의 전달은 중단된다.

그림 9는 3체인 토큰 브리지에서 총 발행 토큰 수를 확인하는 예시를 보여준다. *Pre-Crime*을 사용하기 위해, OApp은 DVN 주소와 *Pre-Crime* 전용 인자를 *메시지 옵션*에 인코딩한다. OApp은 모든 체인의 총 발행 토큰 수 $\sum \text{mint}$ 가 총 잠금 유동성 $\sum \text{lock}$ 이하임을 불변 조건으로 지정한다. 초기에는 $\text{mint}_A = \text{mint}_B = \text{mint}_C = 10\$$, $\text{lock}_A = \text{lock}_B = \text{lock}_C = 10\$$ 이다. 체인 A가 손상되어 아무런 추가 자산을 잠그지 않고 체인 B에서 토큰 10개를 추가 발행하려 시도한다. *Pre-Crime*은 모든 체인의 토큰 수를 확인한 뒤 이를 탐지하여 보안 침해를 단일 체인(체인 A)으로 격리한다. 이후 수신자는 필요하다면 skip(표 3)을 호출하여 해당 논스를 건너뛸 수 있다. 중요한 점은 *Pre-Crime*이 악의적인 DVN이나 블록체인 수준의 결함에 대해 추가적인 작업 증명(proof-of-work)이나 온체인 인프라 부담을 요구하지 않는다는 것이다.

*Pre-Crime*과 *lzCompose*는 *LayerZero*가 지원하는 실행 기능의 두 가지 예시에 불과하다. *메시지 옵션(Message Options)*의 유연성과 검증과 실행의 분리는 *LayerZero*가 다양한 실행 기능으로 확장될 수 있도록 한다.

5 결론

본 논문에서는 *LayerZero* 프로토콜의 설계와 구현을 제시하였다. *LayerZero*는 본질적으로 안전한 크로스체인 메시징과 보편적 의미론을 제공하여, 호환성 그룹 내외의 모든 블록체인을 연결하는 완전 연결 옴니체인 메시 네트워크를 가능하게 한다.

본질적 보안과 외재적 보안을 분리함으로써, *LayerZero*는 채널 무결성의 장기적 안정성을 보장하고 OApp에게 전체 메시 네트워크에 걸친 보편적 네트워크 의미론을 제공한다. *LayerZero*의 보편적 네트워크 의미론과 본질적 보안 보장은 안전한 체인 불가지론적 상호운용성을 가능하게 한다.

새로운 온체인 검증 모듈인 *MessageLib*는 본질적으로 안전한 방식으로 확장 가능한 외재적 보안을 구현한다. 각 OApp은 메시징 채널의 외재적 보안(*MessageLib*, DVN)을 정의하는 *보안 스택(Security Stack)*을 수정할 수 있는 독점적 권한을 가진다. 기존 *MessageLibs*의 불변성은 프로토콜 관리자를 포함한 어떤 주체도 OApp 보안을 일방적으로 훼손할 수 없음을 보장한다.

또한, *LayerZero*는 실행 기능을 패킷 검증으로부터 분리함으로써 보안에 영향을 주지 않고 추가 기능을 구현할 수 있는 사실상 무제한의 자유도를 제공한다. 아울러 *LayerZero*에서 실행과 검증의 분리는 엔지니어링 비용을 절감하고, 버그 발생을 줄이며, 공격 표면을 최소화하고, 전체 프로토콜의 생존성을 향상시킨다.

이러한 구성 요소들이 결합되어, 기존 및 미래의 모든 블록체인에서 보편적 메시징 의미론을 가능하게 하는 고도로 확장 가능한 프로토콜이 완성된다. *LayerZero*는 산발적이고 임시방편적인 메시징 서비스의 환경에 일관성과 단순성을 제공하며, 미래의 완전 연결 옴니체인 메시 네트워크를 위한 토대를 마련한다.

참고문헌

- [1] **Benker, R.** Explained: The nomad hack (August 2022).
<https://www.halborn.com/blog/post/explained-the-nomad-hack-august-2022>
- [2] **Gmytrasiewicz, P. J., and Durfee, E. H.** Decision-theoretic recursive modeling and the coordinated attack problem. In *Proceedings of the First International Conference on Artificial Intelligence Planning Systems* (San Francisco, CA, USA, 1992), Morgan Kaufmann Publishers Inc., p. 88–95.
- [3] **HACXYK.** Wormhole \$10m bounty.
<https://x.com/Hacxyk/status/1529389391818510337>.
- [4] **THORCHAIN.** Post-mortem: Eth router exploits 1 & 2, and premature return to trading incident. <https://medium.com/thorchain/post-mortem-eth-router-exploits-1-2-and-premature-return-to-trading-incident-2908928c5fb>.
- [5] **Zamyatin, A., Al-Bassam, M., Zindros, D., Kokoris-Kogias, E., Moreno-Sanchez, P., Ziogas, A., and Knottenbelt, W. J.** SoK: Communication across distributed ledgers. *Cryptology ePrint Archive*, Paper 2019/1128, 2019.
<https://eprint.iacr.org/2019/1128>.
- [6] **Zarick, R., Pellegrino, B., and Banister, C.** *LayerZero: Trustless omnichain interoperability protocol*.
https://layerzero.network/pdf/LayerZero_Whitepaper_Release.pdf